EXHIBIT 2

13/B
RWB
11-12-03

PATENT
ATTORNEY DOCKET NO. 10491-18

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	McAuley, Aubrey	
Serial No.:	09/410,334	Examiner: John Q. Chavis
Filed:	October 1, 1999	Group Art Unit: 2124
Title:	SYSTEM AND METHOD FOR GENERATING WEB SITES IN AN ARBITRARY OBJECT FRAMEWORK	

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

RESPONSE TO OFFICE ACTION

Sir:

This Response is responsive to the Office Action of April 3, 2003, and is timely submitted with a three-month extension of time.

CONTENTS

Amendments to the Claims are reflected in a listing of all pending claims, which begins on page 2 of this paper.

Remarks begin on page 7 of this paper.

A Conclusion begins on page 17 of this paper.

Attorney Docket No.: 10491-18
Serial No. 09/410,334

AMENDMENTS IN THE CLAIMS

In accordance with 37 C.F.R. § 1.121, please amend the pending claims of the same number according to the following annotated version of all the claims.

B1

1. (Currently amended) A method for generating a computer application on a host system in an arbitrary object framework that separates a content of said computer application, a form of said computer application and a functionality of said computer application, said method comprising;

creating arbitrary objects with corresponding arbitrary names of various object types for generating said content of said computer application, said form of said computer application, and said functionality of said computer application;

managing said arbitrary ~~objections~~ objects in an object library; and

deploying said arbitrary objects from said object library into a design framework to create said computer application.

2. (Original) The method of Claim 1, wherein said computer application is a web site.

3. (Original) The method of Claim 1, wherein said various object types comprise text file pointers.

4. (Original) The method of Claim 1, wherein said various object types comprise binary file pointers.

5. (Original) The method of Claim 1, wherein said various object types comprise compiled executables.

6. (Original) The method of Claim 1, wherein said various object types comprise shell commands.

7. (Original) The method of Claim 1, wherein said various object types comprise remote procedure calls.

8. (Original) The method of Claim 1, wherein said various object types comprise global variables.

9. (Original) The method of Claim 1, wherein said various object types comprise cached executables.

B1

10. (Original) The method of Claim 1, wherein said various object types comprise cached database queries.

11. (Original) The method of Claim 1, wherein said various object types comprise local variables.

12. (Original) The method of Claim 1, wherein said various object types comprise local objects and global parent objects.

13. (Original) The method of Claim 12, wherein said local objects can override said global parent objects.

14. (Original) The method of Claim 12, wherein said local objects inherit data from said global parent objects.

15. (Original) The method of Claim 12, wherein said local objects inherit capabilities from said global parent objects.

16. (Original) The method of Claim 1, further comprising deploying arbitrary objects globally.

17. (Original) The method of Claim 1, further comprising deploying arbitrary objects locally.

18. (Original) The method of Claim l, wherein the step of managing said arbitrary objects further comprises using revision tracking.

19. (Original) The method of Claim 1, wherein the step of managing said arbitrary objects further comprises using rollback.

20. (Original) The method of Claim 1, wherein the step managing further comprises using signoff.

21. (Original) The method of Claim 1, wherein said arbitrary objects can be accessed and deployed into said design framework using said corresponding arbitrary names.

22. (Original) The method of Claim 1, further comprising swapping an arbitrary object of one type with an arbitrary object of another type.

23. (Original) The method of Claim 1, further comprising caching objects.

24. (Original) The method of Claim 23, wherein the step of caching objects further comprises specifying some elements of an arbitrary object to be dynamic elements and specifying some elements of said arbitrary object to be static elements.

B1

25. (Original) The method of Claim 1, further comprising generating arbitrary objects in a programming language that is compatible or supported by said host system.

26. (Original) A method for generating a web site on a host system in an arbitrary object framework that separates a content of said web site, a form of said web site, and a functionality of said web site, said method comprising:

creating arbitrary objects with corresponding arbitrary names of various object types for generating said content of said web site, said form of said web site, and said functionality of said web site; managing said arbitrary objects in an object library; and

deploying said arbitrary objects from said object library to a container page to create said web site.

27. (Original) The method of Claim 26, wherein said various object types comprise text file pointers.

28. (Original) The method of Claim 26, wherein said various object types comprise binary file pointers.

29. (Original) The method of Claim 26, wherein said various object types comprise compiled executables.

30. (Original) The method of Claim 26, wherein said various object types comprise shell commands.

31. (Original) The method of Claim 26, wherein said various object types comprise remote procedure calls.

32. (Original) The method of Claim 26, wherein said various object types comprise global variables.

33. (Original) The method of Claim 26, wherein said various object types comprise local variables.

34. (Original) The method of Claim 26, wherein said various object types comprise local objects and global parent objects.

B1

35. (Original) The method of Claim 34, wherein said local objects can override said global parent objects.

36. (Original) The method of Claim 34, wherein said local objects inherit data from said global parent objects.

37. (Original) The method of Claim 34, wherein said local objects inherit capabilities from said global parent objects.

33. (Original) The method of Claim 26, further comprising deploying arbitrary objects globally.

39. (Original) The method of Claim 26, further comprising deploying arbitrary objects locally.

40. (Original) The method of Claim 26, wherein the step of managing said arbitrary objects further comprises using revision tracking.

41. (Original) The method of Claim 26. wherein the step of managing said arbitrary objects further comprises using rollback.

42. (Original) The method of Claim 26, wherein the step managing said arbitrary objects further comprises using signoff,

43. (Original) The method of Claim 26, wherein said arbitrary objects can be accessed and deployed into said container page using said corresponding arbitrary names.

44. (Original) The method of Claim 26, further comprising swapping an arbitrary object of one type with an arbitrary object of another type.

45. (Original) The method of Claim 26, further comprising caching objects.

46. (Original) The method of Claim 45, wherein the step of caching objects further comprises specifying some elements of an arbitrary object to be dynamic elements and specifying some elements of said arbitrary object to be static elements.

47. (Original) The method of Claim 26, further comprising generating arbitrary objects in a programming language that is compatible or supported by said host system.

B1

48. (Original) The method of Claim 26, wherein said various object types comprise cached executable.

49. (Original) The method of Claim 25, wherein said various object types comprise cached database queries.

50. (Original) The method of Claim 26, further comprising profiling of a user accessing said web site.

51. (Original) The method of Claim 26, further comprising personalization of said web site for a user accessing said web site.

52. (Original) The method of Claim 26, wherein said container page comprises arbitrary objects with both dynamic and static elements.

53. (Original) The method of Claim 26, wherein said content of said web site and said function of said web site can be syndicated.

Background Section of the present application, either alone or in combination, do not suggest the claimed invention, but rather highlight the shortcomings of the prior art that has been overcome by the claimed invention.

Notwithstanding the claim chart for claim 1 in the Office Action, nothing in *Lewandowski* discloses generating a computer application on a host system that "separates a content...a form...and a functionality of a computer application," and "creat[es] arbitrary objects...for generating said content...said form...and said functionality of said computer application," as recited in claim 1. Furthermore, there is no "background acknowledgement" or suggestion in the Background Section of the present application that this feature is disclosed by the *Lewandowski* reference, or any combination of prior art. Respectfully, the Examiner has misconstrued the portion of the Background Section on which he relies to support his position.

In the claim chart entry for the 1st step, the Examiner alleges, "Although, [sic] it is considered inherent that content is a separate object in *Lewandowski's* system, the feature may be argued since is not specifically stated." The Office Action further alleges that the Applicant "admits that the feature existed in the prior art on page 3 lines 18-29 to help minimize dependencies between interface designs and the functions they access."

The Examiner's allegation that a separate object for content is admitted in the Specification is, respectfully, far from accurate. The precise language from page 3 lines 18-29 recites:

> Prior art solutions have succeeded in partially separating some of these functions. Notably, content management databases and digital repositories provide a means of separating content from form and function. Likewise, sophisticated software development teams frequently employ internal code structuring techniques that

> can help to minimize dependencies between interface designs and the functions they access. However, content management tools typically fail to address form/function issues. Therefore, there can still be production slow-downs due to changes in form that require a subsequent change in functionality. (emphasis added).

As can readily be seen from the actual language cited by the Examiner, Applicant does not acknowledge or otherwise admit that creating arbitrary objects for generating separate content, form, and functionality of a computer program is disclosed in the prior art. The cited passage refers to "content management databases," "digital repositories," and "internal code structuring techniques" which have been used in failed attempts to solve the problem of changes in functionality affecting content and form of the computer program, and vice versa. Clearly, Applicant's reference to the failed techniques of the prior art does not admit or imply separating content, form, and functionality, using arbitrary objects in a computer program.

Moreover, separating form, content, and function using arbitrary objects achieves many advantages in accordance with the claimed invention. One such advantage is that when content, form, and functionality are genuinely separated in the generation of a software application, changes in one do not affect the other. See pg. 4, line 24 - pg. 5, line 2.

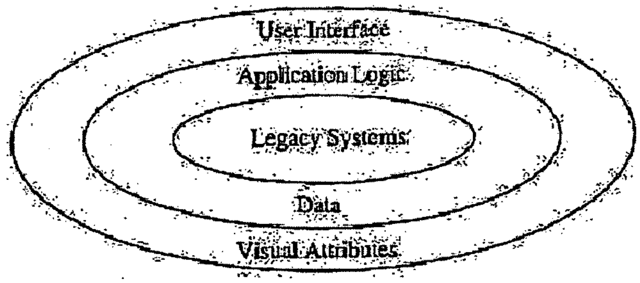
Another advantage to generating arbitrary objects for separating form, content and functionality in a computer program is that users are not required to use a proprietary language to encode the computer program. This allows, for example, the form of systems with legacy content to be updated without affecting the content and functionality of the legacy system. See pg. 5, lines 3-11. Other advantages to generating arbitrary objects for separating these parts of a computer program include allowing single point administrative authority that can reduce security risks, syndication of the software application, and personalization and profiling. See pg. 5, line 12 – pg. 6, line 5.

Further, the *Lewandowski* reference does NOT, in fact, teach creating arbitrary objects for generating separate content, form, and functionality of a computer program. Respectfully, the Examiner misinterprets Figure 10 on page 22 of the *Lewandowski* reference to allege that a business logic object ("BLO") is used to generate content, a presentation object is used to represent form, and a business process object ("BPO") is used to represent functionality. As described on page 22 of the *Lewandowski* reference, the BLO defines how the object reacts to certain events, which clearly is a functional operation, not a content characteristic. Although the BLO is described as storing some business data, the BLO mixes the storing of business data with this functional component. Thus, the BLO of the *Lewandowski* reference still fails to resolve the problems associated with mixing content and functionality in a program. Accordingly, the *Lewandowski* reference still suffers from the same types of shortcomings that are specifically designed to be overcome by the claimed invention.

Moreover, the *Lewandowski* reference plainly states that the "primary difference between a BPO and a BLO is the logical lifetime of the unit of logic: BPOs traditionally handle long-lived processes or processes related to the system as a whole." Thus, to the extent that the BPO is a functional object, so is the BLO, the BLO consisting of shorter-lived processes or processes not necessarily related to the system as a whole.

Lewandowski also states that the three objects are "managed by one object...." As described above, one advantage of the claimed method is that the three object types operate separately from each other, and therefore, changes on one object type would not affect the other object types. In contrast, given that the objects in the *Lewandowski* reference are managed by one object, if the managing object changes, then it will likely be necessary to change the other three objects, and vice versa.

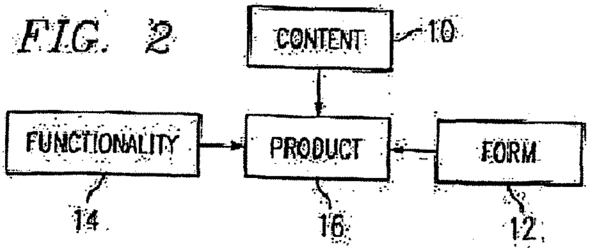
The diagram illustrating *Lewandowski's* application layers in Figure 11 of page 23, is particularly illustrative of *Lewandowski's* failure to separate content and functionality using objects. See below:



The application logic layer, or tier as *Lewandowski* calls it, includes both application logic and data implementation. This is clearly contrary to, and disadvantageous when compared with, the claimed method of the present application. Changes in the format of the data in the *Lewandowski* reference, for example, can clearly affect the functioning of the application logic. Thus, it is abundantly clear that the *Lewandowski* reference has not truly separated content, form, and functionality, using arbitrary objects in a computer program, as required by the claimed invention.

However, Figure 2 of the present application illustrates that the interaction of the content, form, and functionality objects are clearly distinct from the objects utilized in the *Lewandowski* reference.

FIG. 2



Thus, as clarified in the Figure 2, there is no overlap or sharing of layers with the claimed invention of the present application.

In light of the above, the Applicant believes that the 35 U.S.C. §103(a) rejection of claim 1 has been traversed, and claim 1 is in condition for allowance. For the same reasons, Applicant submits that independent claim 26 is in condition for allowance. Claims 2-8, 11-18, 20-21 and 25 depend from claim 1, and claims 27-40, 42-44, 50-51 and 53 depend from claim 26, and therefore, these claims are each in a condition for allowance as well. The Applicant does not herein address the specific rejections for each of those claims because each of those claims depends from one of independent claims 1 and 26, and thus, those claims are allowable based on the allowability of claims 1 and 26. However, the Applicant reserves the right to address the specific rejections of claims 2-8, 11-18, 20-21, 25, 27-40, 42-44, 47, 50-51 and 53 should it be necessary to do so.

5. **Response to Rejections of Claims 9-10, 19, 22-24, 41, 45-46, 48-49 and 52 under 35 U.S.C. §103(a)**

Claims 9-10, 19, 22-24, 41, 45-46, 48-49 and 52 were rejected in the Office Action dated April 3, 2003, under 35 U.S.C. §103(a) as unpatentable over *Lewandowski* in view of purported background acknowledgement of the prior art, and further in view of *Gish* (U.S. Patent No. 6,269,361). Claims 9-10, 19, and 22-24 depend from claim 1. Claims 41, 45-46, and 48-49 depend from claim 26. For brevity, only the bases for the rejection of independent claims 1 and 26 are traversed in detail on the understanding that dependent claims 9-10, 19, 22-24, 41, 45-46, and 48-49 are also patentably distinct over the prior art as they depend directly from claims 1 and 26, respectively. Nevertheless, dependent claims 9-10, 19, 22-24, 41, 45-46, and 48-49 include additional features that, in combination with those of claims 1 and 26, provide further, separate, and independent bases for patentability.

Specifically, the Examiner states that the *Gish* reference teaches "providing caching to speed access time, using rollback to compensate for errors in subclasses via his event handling means, and swapping objects when a different type is required or needed." As with the *Lewandowski* reference, the *Gish* reference does not disclose a method for (1) generating a computer application or web site on a host system that separates content, form, and functionality of the computer application or web site, and (2) creating arbitrary objects for generating the content, form, and functionality thereof, as claimed in claims 1 and 26. Neither does the *Gish* reference fulfill the shortcomings of the Background Section, which is relied upon by the Examiner in combination with the *Lewandowski* reference.

Therefore, since the Applicant has already highlighted the shortcomings of the *Lewandowski* reference in combination with the Background Section of the present application, and the *Gish* reference does not rectify these shortcomings, Applicant submits that the 35 U.S.C. §103(a) rejection of claims 9-10, 19, 22-24, 41, 45-46, 48-49 and 52 has been overcome, and that each of these claims are each in condition for allowance.[1]

[1] Although no arguments have been advanced for the independent patentability of the subject matter of the dependent claims, the Applicant does not acquiesce in the rejection thereof for the reasons advanced in the Office Action, and reserves the right to advance appropriate arguments for patentability of those claims independent of the reasons advanced for patentability of claims 1 and 26 in this or a subsequent proceeding if the patentability or validity of claims 1 or 26 are called into question.

Attorney Docket No.: 10491-18
Serial No. 09/410,334

CONCLUSION

The Applicant has made an earnest and bona fide effort to clarify the issues before the Examiner and to place this case in condition for allowance. In view of the foregoing discussions, it is clear that the differences between the claimed invention and the prior art are such that the claimed invention is patentably distinct over the prior art. Therefore, reconsideration and allowance of all of claims 1-53 are believed to be in order, and an early Notice of Allowance to this effect is respectfully requested. If the Examiner should have any questions concerning the foregoing, the Examiner is invited to telephone the undersigned attorney at (310) 712-8319. The undersigned attorney can normally be reached Monday through Friday from about 9:30 AM to 6:30 PM Pacific Time.

Respectfully submitted,

Dated: October 2, 2003

BROOKE W. QUIST
Reg. No. 45,030
BROWN RAYSMAN MILLSTEIN FELDER
& STEINER LLP
1880 Century Park East, Suite 711
Los Angeles, California 90067
(310) 712-8300